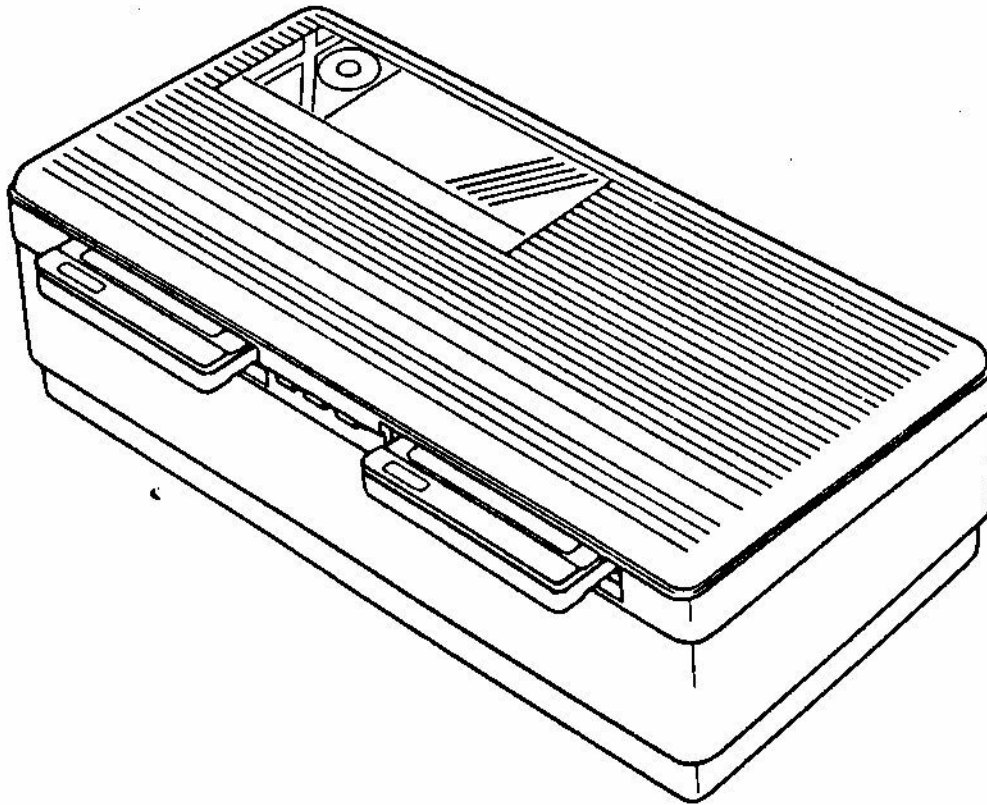


870126; 34 PAGES
FROM HOWARD
JESPERSEN

Wafadrive



ENCLOSED IS A COLLECTION OF TIPS
AND PROGRAMS DESIGNED TO IMPROVE
THE PERFORMANCE OF YOUR WAFADRIVE

COST: \$4.50 INCLUDES
POSTAGE AND UPDATES

Engineering Bulletin

DAMCO

67 Bradley Ct. Fall River, Mass. 02720 - (617) 678-2110

enterprises

INTRODUCTION

THIS MANUAL CONTAINS A COLLECTION OF TIPS AND PROGRAMS DESIGNED TO IMPROVE AND EXPAND THE CAPABILITIES OF THE WAFADrive SYSTEM. THERE IS NO STORAGE SYSTEM CURRENTLY AVAILABLE FOR THE TIMEX/SINCLAIR COMPUTERS THAT OFFER AS MUCH AS THE WAFADrive.

YOUR INITIAL PURCHASE OF THIS MANUAL ENTITLES YOU UPDATE BULLETINS AS THEY BECOME AVAILABLE. IF YOU HAVE ANY INFORMATION THAT YOU WOULD LIKE TO CONTRIBUTE, WE WILL GIVE A FREE WAFER FOR ITEMS PUBLISHED.

CONTENTS

- CHAPTER 1. TAPE TRANSFER UTILITY
- CHAPTER 2. THE FILE UTILITY PROGRAM
- CHAPTER 3. COPY ON EPSON PRINTERS
- CHAPTER 4. CENTRONICS/RS232 CHANNEL DRIVERS (TAB FUNCTION)
- CHAPTER 5. PRINTING GRAPHICS FROM BASIC
- CHAPTER 6. UPGRADING SPECTRAL WRITTER
- CHAPTER 7. CAMPBELL'S MASTER FILE ON WAFADrive
- CHAPTER 8. TASWORD II ON WAFADrive
- CHAPTER 9. VU-FILE & VU-CALC ON WAFADrive

THE CONTENTS OF THIS MANUAL ARE NOT TO BE REPRODUCED WITHOUT WRITTEN PERMISSION FROM DAMCO ENTERPRISES

TAPE TRANSFER UTILITY

When transferring software from cassette to wafer, some kind of transfer utility program is virtually indispensable. The program given here is written largely in BASIC and sets up its own machine code subroutines. The three facilities provided by the program are:

1. READ HEADER

Essential for analysing the attributes of files on cassette. Displays the auto-run line number of BASIC programs, location and length of code files. It works by reading the program header which precedes the file on tape. Beware of false headers!

2. RECLAIM RAM

This option effectively de-initialises the Wafadrive Operating System and reclaims all RAM used. The BASIC program is preserved.

3. STOP

This can be used to load a BASIC program such that it is prevented from auto-running once loaded. It will thus cope with "anti-merge" programs. The auto-run line number is displayed on screen with the program name. The utility program itself is overwritten.

The machine code subroutines used by the program are all relocateable and can be lifted for use in your own programs. Assembly language listings are also given. Memory saving techniques have been used in the BASIC to assist when working in "confined spaces". The program can be further reduced in size if required by splitting it into three parts and/or by loading the machine code straight into the printer buffer as a separate file.

```

10 PAPER VAL "5": INK NOT PI: BORDER VAL "5": GO SUB VAL "1e3"
20 CLS : PRINT "Enter number:"
30 PRINT " 1 READ HEADER" - loads header from cassette
  & displays file attributes."
40 PRINT "' 2 RECLAIM RAM" - de-initialises MOS &
  reclaims RAM."
50 PRINT "' 3 STOP" - loads BASIC program from cassette,
  preventing auto-run."
60 LET a$=INKEY$: IF a$("<1" OR a$(">3" THEN GO TO 60
70 CLS : GO TO VAL "100"*VAL a$
100 REM HEADER READ
110 PRINT " HEADER READ "'Play tape...": RANDOMIZE USR VAL "2
3320": CLS : LET a=VAL "23296"
120 LET t=PEEK a: LET a=a+SGN PI
130 PRINT "TYPE:", "Program" AND (NOT t); "Number array" AND (t=SGN PI);
  "String array" AND (t=VAL "2"); "Bytes" AND (t=INT PI)
140 PRINT "NAME:", : FOR n=SGN PI TO VAL "10": PRINT CHR$ (PEEK
  a); LET a=a+SGN PI: NEXT n
150 PRINT "'LENGTH:", PEEK a+VAL "256*PEEK (a+1)": LET a=a+VAL
  "2"
160 LET b=PEEK a+VAL "256*PEEK (a+1)": IF NOT t THEN PRINT "'AU
  TO-RUN:", : IF B(VAL "1e4" THEN PRINT b
170 IF t=INT PI THEN PRINT "'START:", b
180 PRINT £NOT PI; "Press a key to return to menu": PAUSE NOT PI
: GO TO VAL "20"
200 REM RECLAIM
210 PRINT " RECLAIM RAM "
220 GO SUB VAL "2e3": CLEAR *: RANDOMIZE USR VAL "23340": GO TO
  VAL "20"
300 REM STOP
310 PRINT " STOP "'WARNING! - this program is lost when selec
  ted.": GO SUB VAL "2e3"
320 PRINT "The number shown on screen is the line from which
  the program would normally auto-run." "'Play tape...": USR VAL
  "23369": STOP
1000 RESTORE
1010 DATA "221","33","0","91","221","229","17","17","0","175","5
  5","205","86","5","221","225","48","242","251","201"
1020 DATA "219","12","205","46","10","205","159","0","42","99","
  92","34","101","92","42","146","92","34","104","92","33","182","
  92","1","244","8","195","232","25"
1030 DATA "1","34","0","247","213","221","225","221","54","1","2
  55","221","54","15","128","62","1","50","116","92","205","58","7
  ","62","255","50","68","92","237","75","66","92","205","27","26"
  ,"207","255"
1040 FOR a=VAL "23320" TO VAL "23405": READ a$: POKE a,VAL a$: N
  EXT a: RETURN
2000 REM CONTINUE?
2010 PRINT "'Press Y to proceed or any other key to return to m
  enu"
2020 PAUSE NOT PI: LET a=PEEK VAL "23560": IF a(<>)CODE "Y" AND a(<
  >)CODE "y" THEN RUN
2030 CLS : RETURN

```

```

2  org 60000; De-bug address - normally 23320
3  ;CASSETTE HEADER READER
4  ld ix,23296; Base address
   EA60 DD21005B
ld_hdr:push ix; Save it
   EA64 DDE5
6  ld de,&11; 17 bytes
   EA66 111100
7  xor a; A=0 indicates header
   EA69 AF
8  scf; CY=1 indicates load
   EA6A 37
9  call &0556; Load bytes
   EA6B CD5605
10 pop ix; Retrieve base
   EA6E DDE1
11 jr nc,ld_hdr; Repeat until header loaded
   EA70 30(EA64)
12 ei
   EA72 FB
13 ret
   EA73 C9
14 ;
15 ;
16 ;RECLAIM RAM
reclm:in a,(0ch); page WROM
   EA74 DB0C
18 call clsall; close all channels
   EA76*CD0000
19 call retei; page SROM
   EA79*CD0000
20 ld hl,(stkbot);collapse calc. stack
   EA7C*2A0000
21 ld (stkend),hl
   EA7F*220000
22 ld hl,(membot)
   EA82*2A0000
23 ld (mem),hl
   EA85*220000
24 ld hl,23734
   EA88 21B65C
25 ld bc,2292EA8B 01F40B
26 jp rec_2; reclaim space & return
   EA8E*C30000

clsall:equ 0a2eh
retei:equ 009fh
stkbot:equ 5c63h
stkend:equ 5c65h
membot:equ 5c92h
mem:equ 5c68h
rec_2:equ 19e8h
34 ;
35 ;

```

```

36 ;STOP
stop :ld bc,34; For 2*17-byte headers
      EA91 012200
38 rst &30; Make BC spaces in WKSPCE
      EA94 F7
39 push de; Transfer base to IX
      EA95 D5
40 pop ix EA96 DDE1
41 ld (ix+1),255; Signify null name
      EA98 DD3601FF
42 ld (ix+15),&80; Signify no parameters
      EA9C DD360F80
43 ld a,1 EAA0 3E01
44 ld (tadr_1),a
      EAA2*320000
45 call &073a; Load program
      EAA5 CD3A07
46 ld a,255 EAA8 3EFF
47 ld (nsppc),a; Reset to prevent auto-run
      EAAA*320000
48 ld bc,(newppc); Get line to be jumped to
      EAAD*ED4B0000
49 call &1a1b; Print BC routine in ROM
      EAB1 CD1B1A
50 rst 08; End with error report
      EAB4 CF
51 defb 255 EAB5 FF
tadr_1:equ 23668
nsppc:equ 23620
newppc:equ 23618

```

THE FILE UTILITY PROGRAM

Presented here is an extremely useful utility program for the Wafadrive. It is in the form of a BASIC program which sets up and invokes its own machine code routines. All operations are performed on the wafer in drive A. Most of them work by manipulating the RAM directory - the beauty of a directory driven system (as opposed to the Microdrive for example). Five options are given in the program as follows.

1:RE-NAME FILE

You are asked for the old and new file names, the location of the file entry is found in the RAM directory and the new name "poked" in place of the old. The directory is then updated using a short m/c routine which makes use of a hook code to update the wafer directory. The program sets CAPS LOCK on when requesting the new file name, but if you turn it off, it is possible to re-name the file with lower case characters. This file can then not be loaded etc. from BASIC (try it!). You could use this to implement a simple method of software protection whereby the file could only be loaded back by a special program which converted the RAM directory entry back into upper case.

2:POSITION TAPE

Allows the read/write head to be positioned to any sector along the length of the tape. Note that the "current sector" is the one just read by the Wafadrive. Thus, files can be stored on the wafer with a gap between them such that they chain load optimally. This is achieved by positioning the head to the last sector of the preceeding file.

3:FILE ATTRIBUTES

Probably the most useful option. Effectively a "header reader" which displays the most important attributes of a file including type, designation (which can be different from type if option 5 is used), start address (for BYT files), length and auto-run line number (or address). Also given are the current and directory sectors and a list of the sectors occupied by the file. You will be asked whether you require output to screen or printer (see note below). The program gets the sector information from the directory but the rest is obtained by reading the first sector of the file into the read/write buffer. This contains the actual file attributes which are then "peeked" by BASIC.

4:LAST TO FIRST

The Wafadrive supports the abbreviated command LOAD * which loads the first file appearing in the directory of the default drive. This is intended to be used for BASIC loaders, menu driven programs etc. and can be very useful. The disadvantage is that normally the file entries in the directory are organised chronologically. This means that there is no way of altering the first program without setting it to the end of the directory. However, this option allows the latest file saved to be put to the "top of the heap". It does so by manipulating the RAM directory - moving all the entries down then bringing the bottom one up to the top. As usual, the routine ends by updating the RAM directory to wafer.

5:CHANGE FILE TYPE

Apart from the three file types supported by the WDS (PRG, BYT & DAT), there are five other file types which may appear in the directory - BAK (back-up), TXT (text), LST (list), HLP (help) & ??? (undefined, temporary or unclosed file). The WDS will not recognise any of these files from BASIC, they are intended to be utilised by special programs. Thus you cannot alter a PRG file to LST for example and expect it to load from BASIC (a file type error results). Again, the most likely use of this facility (apart from its novelty value) will be in simple file protection whereupon the file can only be loaded from within a special program which first sets the file type back to normal. At any rate, this option allows you to alter file types at will.

Dedicated "Wafersdrivers" will no doubt find - as I do - the above program indispensable. It is well worth the effort of typing in. Lines 30 & 1395 will need altering to suit different printers. It is set for my own (non-standard) printer as is. If you intend to use a ZX or Alphacom type printer, line 30 can be deleted and the "80" in line 1395 changed to "32".

```

10 REM File utility (P)
20 BORDER 0: PAPER 0: INK 4: CLS
30 CLEAR *: OPEN #3,"C": LPRINT CHR$ 27+"R"+CHR$ 0: REM Initialise printer & select U.S. character set
100 PRINT INK 7;AT 0,9;"MAIN MENU"
101 PRINT AT 5,6;"1 Re-name file"
102 PRINT 'TAB 6;"2 Position tape"
103 PRINT 'TAB 6;"3 File attributes"
104 PRINT 'TAB 6;"4 Last to first"
105 PRINT 'TAB 6;"5 Change file type"
110 PRINT #0;"Enter choice..."
115 LET a$=INKEY$: IF a$>"5" OR a$<"1" THEN GO TO 115
120 CAT #3:"a:": GO SUB 100*(10+VAL A$)
999 RUN
1100 REM Re-name file
1110 CAT #: PRINT " RE-NAME FILE: ""Old name:"",: GO SUB 9700:
GO SUB 9000: PRINT n$;"New name:"",: GO SUB 9700: PRINT n$
1120 FOR x=0 TO 9: POKE a+x,CODE n$(x+1): NEXT x: GO SUB 9600: RETURN
1200 CLS : PRINT " POSITION TAPE: ": GO SUB 9200
1210 PRINT ""Current sector:",PEEK 23769;"Dir. sector:",PEEK 24889;"Enter number of sector to be advanced to (must be in range 1-";PEEK 24885;"): ",
1220 INPUT b: LET b=INT b: IF b<1 OR b>PEEK 24885 THEN GO TO 1230
1230 PRINT b: POKE 23781,b: RANDOMIZE USR 23350
1240 PRINT ""Current sector now: ",PEEK 23769: PAUSE 0: RETURN
1300 CLS : PRINT " FILE ATTRIBUTES: ";AT 7,5;"1 Screen"" 2
Printer";#0;"Enter choice..."
1310 IF INKEY$="2" THEN LET prt=3: GO TO 1330
1320 LET prt=2: IF INKEY$<>"1" THEN GO TO 1310
1330 CAT #: PRINT " FILE ATTRIBUTES: ": GO SUB 9700: CLS : PRINT " FILE ATTRIBUTES: ""#prt;" Filename: ";n$
1340 GO SUB 9000: GO SUB 9900: GO SUB 9300
1345 PRINT #prt;"File designation: ";d$
1350 PRINT #prt;" File type: ";t$
1355 PRINT #prt;" Start: ";s$
1360 PRINT #prt;" Length: ";l$
1365 PRINT #prt;" Auto run: ";a$
1375 PRINT #prt;"Directory sector: ";PEEK 24889
1380 PRINT #prt;" Current sector: ";PEEK 23769
1385 PRINT #prt;" Sector(s) used: ";
1390 LET a=a+11: FOR n=1 TO PEEK a: LET a=a+1: DIM a$(4): LET b$=(STR$ PEEK a)+": LET a$( TO LEN b$)=b$: PRINT #prt;a$;: NEXT n
1395 IF prt=3 THEN FOR n=1 TO 80: LPRINT " ";: NEXT n
1399 PRINT #0;"Press any key": PAUSE 0: RETURN
1400 CAT #: PRINT ""LAST TO FIRST""Press Y to continue or another key to abort this option."
1410 LET a$=INKEY$: IF a$="" THEN GO TO 1410
1420 IF a$<>"y" AND a$<>"Y" THEN RETURN
1430 CLS : PRINT "Updating directory...": GO SUB 9200: RANDOMIZE USR 23296: CAT #: PAUSE 0: RETURN
1500 CAT #: PRINT " CHANGE FILE TYPE: ": GO SUB 9700: GO SUB 9000: GO SUB 9900
1510 CLS : PRINT "Filename:";n$
1520 PRINT AT 3,5;"0 for .PRG file"" 1 for .BYT file""

```

```

2 for .DAT file"" 3 for .BAK file"" 4 for .TXT fi
le"" 5 for .LST file"" 6 for .HLP file"" 7 for
.??? file"" 8 leave unchanged"
1530 PRINT #0;"Current type=";ext;"; enter number of new file ty
pe:";
1540 LET a$=INKEY$: IF a$("<0" OR a$(">8" THEN GO TO 1540
1550 PRINT #0;a$: IF a$="8" THEN RETURN
1560 POKE a+10,ft-ext+VAL a$: GO SUB 9600: RETURN
9000 REM Find file address
9010 LET a=24910: FOR x=1 TO PEEK 24888: LET m$="": FOR y=0 TO 9
: LET m$=m$+CHR$ PEEK (a+y): NEXT y: IF m$=n$ THEN RETURN
9070 LET a=a+12+PEEK (a+11): NEXT x
9080 CLS : PRINT "File:";n$;TAB 19;"...not found.": PAUSE 0: RUN

9200 REM Code poke
9210 RESTORE 9200
9220 DATA 205,69,91,124,181,200,229,25
9222 DATA 35,209,229,183,237,82,193,229
9224 DATA 197,229,193,235,229,229,17,28
9226 DATA 93,237,176,225,17,78,97,183
9228 DATA 237,82,229,193,225,43,209,27
9230 DATA 237,184,193,33,28,93,17,78
9232 DATA 97,237,176,207,39,201,237,91
9234 DATA 229,92,207,43,33,0,0,34
9236 DATA 227,92,207,52,201,33,56,97
9238 DATA 70,4,5,32,4,33,0,0
9240 DATA 201,17,22,0,25,229,17,11
9242 DATA 0,25,126,131,95,225,5,200
9244 DATA 35,24,241,241,0,0,0,0
9246 DATA 30,0,207,45,207,52,201
9250 FOR c=23296 TO 23406
9260 READ b: POKE c,b: NEXT c: RETURN
9300 REM Read file sector
9310 GO SUB 9200: POKE 23401,PEEK (a+12): POKE 23779,0: POKE 237
80,0: RANDOMIZE USR 23400
9320 DEF FN p$(a)=STR$ (256*PEEK (a+1)+PEEK a)
9330 LET t=PEEK 23831: IF t=248 OR t=202 THEN LET t$="PRG"
9332 IF t=2 OR t=3 THEN LET t$="BYT"
9340 LET s$=FN p$(23832) AND t$("<)"PRG": LET l$=FN p$(23834): LET
a$=FN p$(23836) AND (t=3 OR t=202): RETURN
9600 REM Update directory
9610 PRINT #0;"Updating directory...": GO SUB 9200: RANDOMIZE US
R 23347: CAT *: PAUSE 0: RETURN
9700 REM Input file name
9710 POKE 23658,8: INPUT "Enter file name: "; LINE n$: IF LEN n$
>10 THEN PRINT #0;"Filename too long!": PAUSE 50: GO TO 9710
9740 IF LEN n$("<")>10 THEN LET n$=n$+" ": GO TO 9740
9799 RETURN
9900 REM Determine file type
9910 LET ft=PEEK (a+10): LET ext=ft-8*INT (ft/8): LET g=INT (ft/
64)
9930 LET g=(g=2 OR g=3): LET d$="PRGBYTDATBAKTXTLSTHLP???"(3*ext
+1 TO 3*ext+3)
9999 RETURN

```

COPY ON EPSON PRINTERS

The COPY command is not directly supported via the Wafadrive's Centronics or RS-232 interfaces but with a short machine code routine it is possible to implement a high resolution screen dump to printers capable of graphics operation.

Epson control codes are by far the most commonly used and the routine given here utilises these. It will thus also work on many non-Epson printers which recognise ESC "A" and ESC "L".

The routine will give screen dumps in two sizes. It locates in the printer buffer and using the Wafadrive's machine code auto-run facility, self-initialises on loading. Another useful Wafadrive facility is utilised such that the routine is invoked by an additional command DRAW *, which takes the place of COPY (COPY * cannot be used for technical reasons). Note that attempts to LPRINT LLIST or COPY to the Sinclair Printer will cause the code to be wiped from memory.

Two additional "system variables" are created in the printer buffer along with the program. Location 23296 holds the "scale", which can be either 1 or 2 for single or double size (invalid values will be defaulted to 2). The value held in location 23297 determines the left hand margin (spaces). Most printers will ignore excess data if the margin is too wide.

The program given is for the Centronics port (it does not have to be opened to a stream first). To alter for RS-232, POKE 23341,68. The channels do not have to be opened to a stream for the routine to work as it addresses the hardware directly but the baud rate may have to be formatted if not set to 1200.

BREAK can be used to stop printing but has the disadvantage that the printer may well be still in graphics mode when stopped. A better solution is to hold the A key down until printing stops.

```

10 LET a=23296: FOR t=1 TO 15: LET s=0: FOR n=1 TO 16: READ b:
POKE a,b: LET s=s+b: LET a=a+1: NEXT n: READ b: IF b<>s THEN PR
INT "Error in line ";90+10*t: STOP
20 NEXT t: SAVE *"draw dx-85",23296,256,23298: STOP
100 DATA 2,6,33,183,92,1,60,91,113,35,112,219,12,225,195,226,16
05
110 DATA 1,197,197,22,8,120,215,176,34,71,4,126,7,16,253,203,16
50
120 DATA 19,193,4,197,21,32,238,193,193,201,213,95,205,121,26,2
09,2160
130 DATA 201,58,0,91,61,200,62,3,50,0,91,201,254,252,194,26,174
4
140 DATA 1,33,71,91,195,9,1,205,65,10,205,219,1,1,0,0,1107
150 DATA 62,27,205,42,91,62,65,205,42,91,62,8,205,42,91,62,1362
160 DATA 13,205,42,91,205,62,12,40,21,246,32,254,97,32,15,62,14
29
170 DATA 27,205,42,91,62,65,205,42,91,62,12,195,42,91,58,1,1291
180 DATA 91,95,28,29,40,7,62,32,205,42,91,24,246,62,27,205,1286
190 DATA 42,91,62,76,205,42,91,175,205,42,91,205,49,91,58,0,152
5
200 DATA 91,205,42,91,205,17,91,205,49,91,123,40,24,31,31,31,13
67
210 DATA 31,22,4,203,15,203,27,203,43,21,32,247,123,205,42,91,1
512
220 DATA 123,205,42,91,123,205,42,91,12,32,217,62,10,205,42,91,
1593
230 DATA 62,13,205,42,91,205,49,91,62,8,40,2,203,63,128,254,151
8
240 DATA 192,200,71,195,100,91,0,0,0,0,0,0,0,0,0,0,849

```

CENTRONICS/RS-232 CHANNEL DRIVER

When printing to the Wafadrive Centronics or RS-232 ports using LPRINT etc. the TAB command is not normally interpreted correctly. If you have tried using TAB to control an 80-column printer you may have found that the command simply causes the printer to print rubbish or go "haywire".

The reason for this is that TAB is not expanded by the Spectrum's PRINT/LPRINT routines in ROM - it expects the channel driver routine to do it. If it were expanded it would operate on a 32 column basis anyway.

If we require TAB to operate properly we must incorporate our own machine code channel driver to perform the function. Such a routine is described here. Apart from expanding TAB, the program also sets the print width to 64 columns. This gives a margin on 80 column printers and is easier when relating printout to the Spectrum's 32 column screen.

The routine fits easily into the area of memory reserved as a buffer for the ZX Printer - just above the screen. This is not affected when the ZX Printer is not used and most software also leaves it free. The only drawback is that the COPY command will overwrite the contents of the buffer and crash the system next time the printer is accessed.

I must say at this stage that the routine is not my own. It is an adaptation of an RS-232 driver which appears in the Sunshine Publications' book Master Your ZX Microdrive written by Andrew Pennell and reproduced here by his kind permission.

Enter the BASIC program listed overleaf, save and run it. It will generate a copy of the channel driver on your wafer in drive A. If you have an RS-232 printer, before you run the program add the line:

35 POKE 23463,30

The routine is saved as an auto-run machine code file. This is so that it self-initialises on loading. It can subsequently be reset (if you reset your printer for instance) at any time by entering RANDOMIZE USR 23296.

If you use an RS-232 printer the baud rate must be set from BASIC using the FORMAT command. The Centronics version can actually be used without the Wafadrive initialised - if you know how to deinitialise it or if you load the program from cassette.

The principle behind the channel driver is very useful. This routine could be extended to provide page numbering, varying column widths, pause for paper change between sheets, left hand margins etc. The source listing is therefore given for those people conversant with machine code. The assembler used is ZAPP by Hewson Consultants which is available from ROTRONICS.

```

10 REM 64 Column TAB routine
20 CLS : PRINT "Ensure the destination wafer is in drive A wit
h at least 1K spare capacity then press a key:"
30 PAUSE 0: GO SUB 1e3
40 SAVE "tabc 64",23296,176,23296
50 CAT *: STOP
1000 REM DATA
1010 DATA 42,79,92,1,15,0,9,1,35,91,113,35,112,175,33,171,1004
1020 DATA 91,119,35,119,61,253,119,0,30,3,207,35,237,123,61,92,1
585
1030 DATA 195,118,27,33,172,91,203,78,40,40,203,86,40,5,203,142,
1676
1040 DATA 203,150,201,203,214,87,214,64,210,108,4,43,122,150,35,
220,2228
1050 DATA 102,91,122,43,150,35,200,71,62,32,197,217,215,217,193,
16,1963
1060 DATA 247,201,254,32,48,31,254,13,40,7,254,23,63,192,203,206
,2068
1070 DATA 201,203,70,203,134,192,203,134,43,54,0,35,62,10,205,16
3,1912
1080 DATA 91,62,13,24,46,254,165,56,5,214,165,195,16,12,203,134,
1655
1090 DATA 33,59,92,203,134,254,32,32,2,203,198,254,128,56,2,62,1
744
1100 DATA 63,205,163,91,33,171,91,52,126,254,64,192,35,205,102,9
1,1938
1110 DATA 203,198,201,213,229,95,207,31,225,209,201,0,0,0,0,0,20
12
1200 CLS : PRINT "Wait a while..."
1210 RESTORE 1e3: LET a=23296: FOR m=1 TO 11: LET s=0: FOR n=1 T
O 16: READ b: POKE a,b: LET s=s+b: LET a=a+1: NEXT n: READ b: IF
b<>s THEN PRINT "CHECKSUM ERROR IN LINE ";1e3+10*m: STOP
1220 NEXT m: RETURN

```

PRINTING GRAPHICS FROM BASIC

The Spectrum LLIST and LPRINT commands automatically expand the character codes outside the normal ASCII range 0-127 into Spectrum tokens - see appendix A of the Spectrum BASIC Programming Manual. Codes above 165 are expanded into commands and symbol tokens which are ASCII composites, comprising anywhere between two (PI, FN, AT etc.) and nine (RANDOMIZE) characters.

Most printers use control codes which lie in the range 0-127. So there is usually no problem in accessing facilities such as underlining, bold printing etc. from BASIC. The problem comes when attempting to print 8-pin graphic data. When the most significant bit (msb) of the data is set (ie. codes above 128), the Spectrum expands the data into a string of codes before transmitting it to the RS-232 or Centronics ports for printing.

The result of this is twofold. Firstly, the intended print pattern is not reproduced and the graphics are distorted. Secondly, more codes are sent than intended and the printer invariably gets confused, printing random characters etc.

There are two solutions:

1. The graphics data can be restricted to 7 bits or less and the line feed increments reduced to suit. This is usually possible but it can make the BASIC complicated and the printing slower.
2. Resort to machine code. A simple routine is all that is required to pass the data directly to the Centronics or RS-232 port and this can be incorporated into a BASIC routine very easily thus:

```
10 DATA 30,0,207,31,201
20 LET A=USR "A": FOR C=A TO A+4: READ N: POKE C,N: NEXT C
```

This locates in the UDB area but can be relocated anywhere by altering the value of A. To send a byte to the printer use:

```
POKE A+1,N: RANDOMIZE USR A
```

Where N is the byte to be sent. For RS-232, simply alter the number 31 in line 10 to 30. Note that the channel does not have to be previously opened to a stream for the routine to work.

For the interested the machine code is as follows:

```
LD E,N      ;N POKED IN FROM BASIC
RST 08      ;FORCE AN ERROR - PAGE IN WROM
DEFB 31     ;SIGNAL "O/P TO CENTRONICS" (DEFB 30 FOR RS-232)
RET         ;BACK TO BASIC
```

For example, a routine to print a "chequerboard" pattern on an Epson printer can be achieved by adding the following lines to the BASIC above:

```
30 DATA 27,75,8,0: REM ESC,K,n,n
40 DATA 170,85,170,85,170,85,170,85: REM GRAPHICS DATA
50 FOR C=1 TO 12: READ N: POKE A+1,N: RANDOMIZE USR A: NEXT C: STOP
```

UPGRADING SPECTRAL WRITER

Spectral Writer is an extremely useful program but with a few modifications it can be improved even further. I use the program all the time and over a period have evolved a number of "bolt-on goodies" which I present here in the form of an upgrade to "version 3.0".

1: KNOWN BUGS

There are two known "bugs" in the original program - both of which can have serious consequences unless you know about them. Firstly, the EM[Y] command which is frequently used during editing to split a line should not be used with the cursor in column 63 - ie. at the end of a line. For reasons which have not (yet) been identified this causes a system reset. Secondly, the mechanism which automatically returns you to the text file when a faulty parameter is entered does not work properly. The next time the block option menu is used to select a function, the program "hangs". Fortunately there is something we can do about this - in version 3.0 the feature is simply removed. If any errors cause the program to stop with an error report simply enter RUN to get back to the text file.

2: DIRECTORY DISPLAY

It is tiresome to have to select a separate option (7) to view the directory before saving, loading or erasing files. Version 3.0 asks for the required drive then displays the wafer directory before you are required to enter the file name. This makes these operations much more convenient and pleasant to use.

3: MERGING TEXT FILES

You may have observed that whilst it is possible to merge (or more correctly append) text files when loading from cassette, it is not possible to do so from wafer. Again, version 3.0 puts this right - but there is a limitation. If the total text loaded exceeds the overall text capacity of 352 lines then the program code will be overwritten - so take care. It is possible to combine two files that exceed the text capacity by inserting the statement LOAD *a:code" before RUN in line 1620. The program code will then load from the program wafer (which must be in drive A) and overwrite any text which overlaps. I use this technique quite often when juggling with multi-file documents.

4: PRINTING OF MULTIPLE COPIES

Without access to a photo-copier it is often convenient to be able to leave the system printing out a document repeatedly. Some of you may even have got as far as identifying the print routine in the BASIC section and tried incorporating it in a FOR NEXT loop in an effort to achieve this. The problem is that

the print routine leaves the cursor at the bottom of the text file when it is finished, and since printing always takes place from the current cursor position, none but the first copy gets printed. Version 3.0 overcomes this by resetting the cursor pointer after each print.

If using the multiple print facility you will probably have to define a graphic control character for Form Feed (usually 12) and put it at the end of the text file so that printing always restarts from a new page.

If page numbering is used, it will increment all the way through the copies. It can be reset by adding the statement `POKE VAL "55533", VAL "n"` among the pokes in line 3260 - n being the desired start page number.

5: DEFINING GRAPHICS CONTROL CHARACTERS

The problem when entering these control characters, is that the screen clears whilst the codes are being input. Thus it is quite easy to "lose track" of what you are doing. In version 3.0 the list of characters and their codes is kept on screen whilst new ones are being entered.

6: USING SINGLE SHEET PAPER

Printing of multi-page documents on single sheets of paper can be very awkward indeed. Some printers are equipped with paper end detection which deselects the printer and allows a new sheet of paper to be inserted, but in a lot of cases this is designed to function as an "early warning" with fanfold paper and operates 2 or 3 inches before the end of the sheet is reached. Version 3.0 offers you the option of a pause between pages to allow paper to be changed.

When this is selected, printing will automatically be suspended until a key is pressed on the keyboard. No prompt message is displayed so as to keep the modifications as simple as possible.

7: PARAMETER SELECT AT PRINT TIME

It can be a nuisance to have to select option A from the block option menu to check things like page numbering before printing. Version 3.0 prompts for interface code, baud rate (if interface code not zero), page numbering mode, next page number (if numbering mode not zero), lines per page, margin, number of copies and pause for paper change when option 8 is selected.

8: SPEEDING UP LOADING

If you use Spectral Writer often it is desirable that it loads as quickly as possible. The backup option in version 3.0 saves the three program files in such a way as to minimise loading time - it is now typically 20 secs rather than 30. The principle


```
POKE 31515,1      POKE 31516,174    POKE 31517,253
POKE 31518,203    POKE 31519,1      POKE 31520,110
POKE 31521,192    POKE 31522,24     POKE 31523,249
                (This is the "paper pause" routine)
```

5. Insert the blank 16K wafer in drive A then enter GO TO 7950 to duplicate the new version. It is also a good idea to make a backup of it onto another wafer by entering GO SUB 9000 (which makes a copy without formatting the wafer).

Version 3.0 will overwrite any files of the same name when saving text. I find this more convenient than having to erase old versions of documents before I save the new ones. It is of course a bit more "risky" in that you could accidentally overwrite an important file. If you want to safeguard against this modify line 2000 to use SAVE * rather than SAVE #.

Using standard 11" fanfold paper you will get 66 lines at the normal pitch of 6 lpi. My printer has a "perf skip" facility which leaves a 1" (6 line) gap between forms. I tend to leave this feature switched in as it is useful when printing listings etc. and thus I suppress Spectral Writer's pagination by setting the Form Feed code to zero. This leaves 60 lines per page - or 58 if the page numbering facility is selected. A4 paper gives about 70 lines overall.

Finally, a few words about graphics control codes. My own version is set up as follows:

CHR\$ CODE	FUNCTION
143	U.K. Character set
128	U.S. " "
142	Enlarged printing 'on'
129	" " 'off'
141	Condensed " 'on'
130	" " 'off'
140	Underlining 'on'
131	" 'off'
139	Italics 'on'
132	" 'off'
138	Emphasised printing 'on'
133	" " 'off'
137	Enhanced " 'on'
134	" " 'off'
136	Elite printing 'on'
135	" " 'off'

```

1 REM (c) 1984 Gianluca Carri - VSN. 3.0
10 POKE VAL "23658",NOT PI: GO TO VAL "110"
100 IF s THEN CLEAR VAL "31505"
105 PRINT AT VAL "18",NOT PI:; LOAD "CODE 1"CODE : PRINT AT VAL
"18",NOT PI:; LOAD "CODE "CODE
107 RANDOMIZE USR VAL "60063": RANDOMIZE USR VAL "31525"
110 RANDOMIZE USR VAL "55363"
120 LET wd=VAL "9520": LET yn=VAL "32332": LET dc=VAL "256": LE
T x=PEEK VAL "55496"+dc*PEEK VAL "55497": LET p=VAL "9800": LET
fn=VAL "9050": LET pr=VAL "8e3": LET def=VAL "7900": LET z=SGN P
I: BRIGHT NOT PI
121 LET a=PEEK VAL "23613"+dc*PEEK VAL "23614": POKE a,INT PI:
POKE a+SGN PI,VAL "19"
130 LET c$=CHR$ USR VAL "31541": IF USR yn THEN GO TO VAL "130"
140 INPUT "": IF c$>"9" THEN GO TO (CODE c$-VAL "55")*VAL "400"
150 GO TO VAL c$*VAL "400"
400 CLS : LET a$=CHR$ VAL "2"+"COPY BLOCKS OF TEXT": GO SUB pr:
LET a$=CHR$ VAL "5"+"Beginning line? ": GO SUB pr: INPUT s: LET
a$=a$+STR$ s: GO SUB pr: LET a$=CHR$ VAL "7"+"Ending line? ": G
O SUB pr: INPUT e: LET a$=a$+STR$ e: GO SUB pr: LET a$=CHR$ VAL
"9"+"New beginning line? ": GO SUB pr: INPUT n: LET a$=a$+STR$ n
: GO SUB pr: LET l=e-s+SGN PI: IF n+1>VAL "351" THEN LET a$=CHR$
VAL "12"+"OUT OF FILE ERROR": GO TO VAL "435"
410 LET l=l*VAL "64": LET s=VAL "s*64+32768": LET n=VAL "n*64+3
2768": IF n>s THEN GO TO VAL "440"
420 LET s=s+1-SGN PI: LET n=n+1-SGN PI: IF n<s-1 THEN RANDOMIZE
1+n/(s/USR VAL "32551"): RUN
430 LET a$=CHR$ VAL "12"+"OVERLAPPING ERROR"
435 GO SUB pr: FOR x=NOT PI TO VAL "100": NEXT x: RUN
440 IF n<s+1 THEN GO TO VAL "430"
450 RANDOMIZE 1+n/(s/USR VAL "32526"): RUN
800 CLS : LET a$=CHR$ VAL "2"+"LOAD/MERGE TEXT FROM TAPE": GO S
UB pr: LET z=NOT PI: GO SUB fn: IF NOT LEN f$ THEN POKE VAL "232
96",-SGN PI: GO TO VAL "820"
810 FOR g=SGN PI TO VAL "10": IF g>LEN f$ THEN POKE VAL "23295"
+g,VAL "32": NEXT g
815 POKE VAL "23295"+g,CODE f$(g): NEXT g
820 GO SUB VAL "7850": IF USR yn THEN RANDOMIZE USR VAL "32568"
: RUN
830 RANDOMIZE USR VAL "60063": LOAD f$CODE VAL "32768": RUN
1200 CLS : LET a$=CHR$ VAL "2"+"SAVE TEXT ON TAPE": GO SUB pr: G
O SUB fn: GO SUB VAL "9600": SAVE f$CODE st,ln: GO SUB VAL "9510
": IF NOT USR yn THEN VERIFY f$CODE st,ln
1210 RUN
1600 CLS : LET a$=CHR$ VAL "2"+"LOAD TEXT FROM WAFADRIE": GO SU
B pr: GO SUB wd: GO SUB fn: LET st=VAL "32768": GO SUB VAL "7850
": IF USR yn THEN LET st=st+VAL "64*(1+INT (x/64))": GO TO VAL "
1620"
1610 RANDOMIZE USR VAL "60063"
1620 LOAD *d$+f$,st: RUN
2000 CLS : LET a$=CHR$ VAL "2"+"SAVE TEXT ON WAFADRIE": GO SUB
pr: GO SUB wd: GO SUB fn: GO SUB VAL "9600": SAVE *d$+f$,st,ln:
GO SUB VAL "9510": IF NOT USR yn THEN VERIFY *d$+f$
2010 RUN
2400 CLS : LET a$=CHR$ VAL "2"+"ERASE FILE ON WAFADRIE": GO SUB
pr: GO SUB wd: GO SUB fn: LET a$=CHR$ VAL "7"+"Press a key to e
rase the file": GO SUB pr: PAUSE NOT PI: ERASE *d$+f$: RUN
2800 CLS : LET a$=CHR$ VAL "3"+"DIRECTORY OF WAFER": GO SUB pr:
GO SUB wd: PAUSE NOT PI: RUN

```

```

3200 CLS : LET a$=CHR$ VAL "2"+"PRINT OUT THE TEXT": GO SUB pr:
LET n=VAL "5": LET a$="Interface code": LET a=VAL "55492": GO SU
B p: IF NOT v1 THEN GO TO VAL "3240"
3210 LET a$=CHR$ n+"Baud rate? ("+STR$ (VAL "PEEK 64460+dc*PEEK
64461")+"): ": GO SUB pr: GO SUB def: IF v1=dc THEN LET v1=VAL "
PEEK 64460+dc*PEEK 64461"
3220 IF v1<NOT PI OR v1>VAL "9600" THEN GO TO VAL "3210"
3230 POKE VAL "64460",v1-dc*INT (v1/dc): POKE VAL "64461",INT (v
1/dc): LET a$=a$+STR$ v1: GO SUB pr: FORMAT *"r",v1: LET n=n+SGN
PI
3240 LET a$="Page numbering mode": LET a=VAL "55499": GO SUB p:
IF v1 THEN LET a$="Next page number": LET a=VAL "55533": GO SUB
p
3250 LET a$="Lines per page": LET a=VAL "55532": GO SUB p: LET a
$="Left margin": LET a=VAL "55495": GO SUB p: LET a$=CHR$ n+"How
many copies? (1): ": GO SUB pr: GO SUB def: IF v1=dc THEN LET v1
=SGN PI
3255 LET n=n+SGN PI: LET a$=CHR$ n+"Pause for paper change? (y/n
)": ": GO SUB pr: POKE VAL "31510",USR yn
3260 CLS : FOR a=SGN PI TO v1: RANDOMIZE USR VAL "55652": POKE V
AL "55601",NOT PI: POKE VAL "55602",NOT PI: NEXT a: RUN
3600 CLS : RANDOMIZE USR VAL "32355": IF USR yn THEN GO TO VAL "
3650"
3610 PRINT AT VAL "17",NOT PI;" " : LET a$=CHR$ V
AL "17"+"Insert character code: ": GO SUB pr: INPUT ch: IF ch<VA
L "128" OR ch>VAL "143" THEN GO TO VAL "3600"
3620 LET a$=a$+STR$ ch: GO SUB pr
3630 INPUT "code1: ";c1,"code2: ";c2,"code3: ";c3,"code4: ";c4: IF c
1>=dc OR c2>=dc OR c3>=dc OR c4>=dc OR c1<NOT PI OR c2<NOT PI OR
c3<NOT PI OR c4<NOT PI THEN GO TO VAL "3630"
3640 LET x=VAL "(ch-128)*4+64558": POKE x,c1: POKE x+SGN PI,c2:
POKE x+VAL "2",c3: POKE x+INT PI,c4: GO TO VAL "3600"
3650 CLS : LET n=SGN PI: LET a$="Carriage return": LET a=VAL "55
493": GO SUB p: LET a$="Linefeed": LET a=VAL "55494": GO SUB p:
LET a$="Formfeed": LET a=VAL "55498": GO SUB p: RUN
4000 CLS : LET a$=CHR$ VAL "2"+"REDEFINE PARAMETERS": GO SUB pr:
LET n=VAL "5": LET a$="Column at which the bell rings": LET a=V
AL "55490": GO SUB p
4010 LET a$="Length of keyboard click": LET a=VAL "23609": GO SU
B p: LET a$="Delay before auto-repeat": LET a=VAL "23561": GO SU
B p: LET a$="Speed of auto-repeat": LET a=VAL "23562": GO SUB p
4060 LET a$=CHR$ n+"Page header?": GO SUB pr: LET a$=CHR$ VAL "n
+2": FOR a=SGN PI TO VAL "32": LET a$=a$+CHR$ PEEK (a+55499): NE
XT a: GO SUB pr: INPUT LINE a$: IF NOT LEN a$ THEN RUN
4070 IF LEN a$>VAL "32" THEN GO TO VAL "4060"
4080 RANDOMIZE USR VAL "60089": DIM n$(VAL "32"): LET n$=a$: FOR
a=SGN PI TO VAL "32": POKE VAL "55499"+a,CODE n$(a): NEXT a: RU
N
4400 CLS : CLEAR : LET s=SGN PI: IF USR VAL "32692" THEN SAVE "-
writer" LINE VAL "100": SAVE "CODE 1"CODE VAL "31525",VAL "1243"
: SAVE "CODE "CODE VAL "55360",VAL "10176": PRINT "REWIND TAPE &
PRESS A KEY TO""VERIFY": PAUSE NOT PI: VERIFY "": VERIFY "COD
E : VERIFY "CODE : RUN
4410 GO SUB VAL "9e3": RUN
4800 POKE VAL "23658",NOT PI: RANDOMIZE USR VAL "57623"
5200 RANDOMIZE USR VAL "59275"
7850 CLS : LET a$=CHR$ VAL "2"+"Load (y) or Merge (n)?": GO SUB
pr: RETURN
7900 INPUT LINE b$: IF b$="" THEN LET v1=dc: RETURN
7910 LET v1=VAL b$: RETURN

```

```

7950 FORMAT *"

```

CAMPBELL'S MASTERFILE ON THE ROTRONICS WAFADRIVE

These notes, and accompanying listings, are for the benefit of users of Campbell's Masterfile program wishing to convert it for use on the ROTRONICS Wafadrive.

The Wafadrive has no direct SAVE..DATA function, so the method used here saves the program and data together. The program is short so the speed penalty is minimal. The loader program shown allows you to select which file you require to begin with. A nice feature of this is that upon entering a null file name (by just pressing ENTER), the last file accessed will be loaded. The conversion procedure is as follows:

Format a fresh (16K) wafer and insert it in drive A. Type in the loader program and save it using:

```
SAVE *"MF LOAD" LINE 1000: VERIFY *: CAT *
```

The VERIFY and CAT commands have the effect of positioning the tape to minimise loading time - don't omit them.

Load in an "empty" version of Masterfile from cassette, just as you would normally. Do not reset the computer or type NEW first. Exit to BASIC (see MASTERFILE handbook), then modify the program as shown. The changes are extensive so check the results carefully. You can start from fresh but you must restore MASTERFILE's reserved variables. When done, execute:

```
SAVE *"MF" LINE 1: VERIFY *"MF": CAT *  
SAVE *"MF MCODE",57328,8207: VERIFY *"MF MCODE"
```

Never RUN/CLEAR the BASIC program - otherwise all your data will be lost. If the program stops with an error report ("wafer not inserted" etc.), restart by entering GO TO 1. Do this now and save the new version of the program onto your back-up wafer (option V from the MASTERFILE main menu).

You will be able to load your old cassette files using LOAD ** DATA f\$() as per usual. They can then be saved onto the Wafadrive from your new version of the MASTERFILE program. If you have MF PRINT, this too can be loaded in from cassette. Once loaded it remains imbedded in the Masterfile program.

MASTERFILE normally interfaces with printers via the COPY command. If you plan to use a full-size printer with the Wafadrive's Centronics or RS-232 ports, you will need some software to simulate COPY. Such routines are available for EPSON compatible printers (contact ROTRONICS for details).

Finally, these notes are given for guidance only and refer to version 09 of Masterfile. The changes required for version 08 are also shown and I am indebted to Mr. C. Hurst of Maidenhead for this information. I would be pleased to hear from anyone else who has any further suggestions or comments.

G.P. BOOTH - TECHNICAL DIRECTOR, ROTRONICS LTD.

May 1985

```
10 REM MASTERFILE LOADER  
20 PRINT AT 3,5;"CAMPBELL'S MASTERFILE";AT 5,7;"(Wafadrive ve  
rsion)"  
30 PRINT #0;"Drive from which file is to be loaded (A/B):"  
40 LET d$=INKEY$: IF d$="" THEN GO TO 40  
50 LET d$="b:" AND (d$="B" OR d$="b"): CAT *d$  
60 POKE 23658,8: DIM n$(10): INPUT "File name?", LINE n$
```

```

100 REM SEARCH FOR FILE
110 LET a=24910+582*(d$="b:")
120 FOR x=1 TO PEEK (a-22): LET m$=""
130 FOR y=0 TO 9: LET m$=m$+CHR$ PEEK (a+y): NEXT y
140 IF m$=n$ THEN GO TO 200
150 LET a=a+12+PEEK (a+11): NEXT x
160 IF n$<>" " THEN PRINT #0;"FILE NOT FOUND": PAUSE 0
: GO TO 60
200 LOAD *d$+m$
210 PRINT #0; FLASH 1;"ERROR - WRONG FILE TYPE": PAUSE 0: STOP
1000 CLEAR 57327: CLS *: LOAD *"mf mcode": POKE 65535,24: RUN

```

Masterfile BASIC program:

```

1 PRINT ;: PAPER VAL "7": GO TO USR VAL "58285"
2 LOAD *"mfp util"
3 LOAD *"mfp ovly": GO TO SGN PI
4 LOAD *"mf mcode": GO TO SGN PI
5 SAVE *"mf mcode",VAL "57328",VAL "8207": GO TO SGN PI
50 PRINT AT VAL "15",VAL "2";"Cassette(c) or W/disk(a/b)"
60 POKE VAL "23658",NOT PI: LET d$=INKEY$: IF d$<"a" OR d$>"c"
THEN GO TO VAL "60"
65 LET d=NOT d$="c": LET d$=d$+":": BEEP VAL ".1",NOT PI: IF d
THEN CAT *d$
66 RETURN
70 PRINT AT VAL "17",VAL "2";"Verify ? Y/N"
75 IF INKEY$="Y" OR INKEY$="y" THEN BEEP VAL ".1",NOT PI: PRIN
T AT VAL "17",VAL "8": BRIGHT SGN PI;"ing...": GO TO VAL "90"
80 IF INKEY$="N" OR INKEY$="n" THEN : RETURN
85 GO TO VAL "75"
90 IF d THEN VERIFY *d$+n$: GO TO USR r
95 VERIFY n$: GO TO USR r
4000 INPUT PAPER VAL "7";(c$( TO VAL "32")): LINE c$: GO TO USR
r
4010 COPY : GO TO USR r
4020 LET n$=c$( TO VAL "10"): GO SUB VAL "50": IF NOT d THEN SAV
E n$ LINE SGN PI: GO SUB VAL "70": GO TO USR r
4025 SAVE *d$+n$ LINE SGN PI: GO TO USR r
4030 CLS *: PRINT "PROGRAM BACK-UP""Ensure master wafer in A,
blank in B - then press a key."': PAUSE NOT PI: MOVE *"a:mfp loa
d" TO "b:": SAVE *"b:mfp" LINE SGN PI: SAVE *"b:mfp mcode",VAL "57
328",VAL "8207": GO TO USR r
4040 LET c$=STR$ (VAL c$( TO VAL "14")/VAL c$(VAL "15" TO VAL "1
9")): GO TO USR r
4050 LET n$=c$( TO VAL "10"): GO SUB VAL "50": IF d THEN LOAD *d
$+n$: GO TO USR r
4055 PRINT "Play the tape...": LOAD n$: GO TO USR r
4200 PRINT AT VAL "13",NOT PI: BRIGHT SGN PI: PAPER VAL "6";"Giv
e name of print spec to load": INPUT n$: IF n$="" THEN GO TO SGN
PI
4202 PRINT "Play the tape...": LOAD n$ DATA p$(): GO TO USR r
4210 REM See MFP User Guide
4290 CLOSE #*INT PI: GO TO USR r
7000 LET c$="": GO TO USR r
9000 BEEP SGN PI,NOT PI: GO TO USR r

```

Modifications for version 08:

```

1 PRINT ;: PAPER VAL "7": GO TO USR VAL "64951"
5 SAVE *"mf mcode",VAL "57037",VAL "8499": GO TO SGN PI
4030 CLS *: PRINT "PROGRAM BACK-UP""Ensure master wafer in A,
blank in B - then press a key."': PAUSE NOT PI: MOVE *"a:mfpload
" TO "b:": SAVE *"b:mfp" LINE SGN PI: SAVE *"b:mfp mcode",VAL "570
37",VAL "8499": GO TO USR r

```

TASWORD II ON WAFADRIVE

Although the Wafadrive is normally supplied with an excellent Word Processor program (Spectral Writer), there are people who already own Tasword II and who wish to transfer it to wafer.

This is understandable - it takes some time to become familiar with a complex program such as a word processor and once experience is gained, many people are reluctant to start again with another program. Also, Tasword does have some features which Spectral Writer lacks (although the opposite is also true).

It should be noted however that Spectral Writer and Tasword text files are interchangeable - albeit with a small modification to Spectral Writer BASIC. (For those who are interested, amend the statement `LOAD *f$` in line 1600 to `LOAD *f$,VAL 32768`).

The fundamental problem with Tasword is that the extra RAM reserved by the Wafadrive causes there to be insufficient room below RAMTOP for the BASIC. There are two solutions: the "official" TASMAN solution, (details of which can be obtained by writing to them), involves raising RAMTOP and reducing the program's text capacity by 24 lines.

Tasword's BASIC portion is particularly space-inefficient and a better solution is to condense it so that it will fit into the reduced space available. The only real problem with this is that the whole program must be edited through. However, it is well worth the hour or so's effort.

So, to convert Tasword start by loading the program from cassette in the normal way with the Wafadrive connected but not initialised. Exit to BASIC and go through every line of the program such that the final result appears exactly as per the listing below. Do not make any other changes at this stage. One small point - there should be a copyright symbol immediately before the word "Tasman" in line 4000. Since this is a non-ASCII character, most printers will not reproduce it.

When finished, you should find that you are able to enter `NEW *` to initialise the Wafadrive. Do so and save the BASIC on a spare wafer. Now you can run the program and test it. If all appears well use Tasword's backup option to save the new version onto a blank 16K wafer.

As shown the program provides for a printer connected via the Wafadrive's Centronics port. If you have an RS-232 printer, change the "c" in line 15's `OPEN #` statement to "r". You will also need to add a `FORMAT` statement after it if the baud rate is anything other than 1200.

Your old cassette files can be loaded by temporarily altering line 2010 back to cassette load syntax.

For those people not familiar with memory-saving techniques on the Spectrum it may seem strange to replace a "straight" number with its string equivalent and use VAL. However, because of the way the Spectrum stores numbers 3 bytes are saved every time this is done. Using things like NOT PI (0), SGN PI (1) and INT PI (3) saves 5 bytes every time. The savings may seem small but these and other techniques lop over 1.5K from the original version of the program. (In fact, by re-writing the BASIC completely it can be reduced even further but the result was considered too radically different to present here). The only penalty of the changes is that the functions carried out by Tasword's BASIC become marginally more sluggish in operation. In practise this is a minor irritation.

TASWORD BASIC LISTING (WAFADRIE VERSION):

```

10 CLS : LET a=USR VAL "64330": GO TO VAL "20"
15 POKE VAL "23609",VAL "2": CLEAR VAL "31999": GO SUB VAL "4e
3": LOAD *"tascode": RANDOMIZE USR VAL "59081": OPEN #*INT PI,"c
": GO TO VAL "10"
20 CLS : LET a=VAL "64*INT (a/64+.99)": IF NOT a THEN GO TO VA
L "3e3"
25 GO SUB VAL "4e3": PRINT "drive directory"
26 PRINT "print text file"
28 PRINT "save text file"
30 PRINT "load text file"
35 PRINT "merge text file"
40 PRINT "return to text file"
45 PRINT "define graphics/prINTER"
50 PRINT "save tasword"
55 PRINT "into Basic"
60 PRINT "erase file"
70 INPUT "": PRINT #NOT PI;AT SGN PI,VAL "10": INVERSE SGN PI;
"press key": GO SUB VAL "8200"
80 LET a$=INKEY$: IF NOT LEN a$ THEN GO TO VAL "80"
90 LET i=NOT PI: LET b=CODE a$: LET b=b+VAL "32"*(b<VAL "97")
110 IF b=VAL "115" THEN LET i=VAL "6"
120 IF b=VAL "106" THEN LET i=VAL "8"
125 IF b=VAL "116" THEN LET i=VAL "16"
130 IF b=VAL "112" THEN LET i=VAL "4"
140 IF b=VAL "121" THEN LET i=VAL "12"
150 IF b=VAL "109" THEN LET i=VAL "10"
160 IF b=VAL "103" THEN LET i=VAL "14"
170 IF b=VAL "98" THEN LET i=VAL "18"
175 IF b=VAL "100" THEN LET i=VAL "2"
178 IF b=VAL "101" THEN LET i=VAL "20"
180 IF i THEN PRINT AT i,VAL "31": FLASH SGN PI;CHR$ b: GO TO
VAL "500"
190 GO TO VAL "80"
200 CLS : PRINT AT VAL "4",VAL "8":"PRINT OPTIONS"" just pres
s ENTER for default values given in brackets"
210 LET i=VAL "8": LET j0=VAL "23": PRINT AT i,NOT PI,"Line spa
cing? (1)": GO SUB VAL "6e3": IF NOT LEN a$ THEN LET a$="1"
215 POKE VAL "62235",VAL a$

```

```

220 LET i=VAL "10": PRINT AT i,NOT PI;"Start at line? (1)": GO
SUB VAL "6e3": IF NOT LEN a$ THEN LET a$="1"
230 LET c=VAL "64*(INT VAL a$-1)": LET b=c+FN p(VAL "62216"): L
ET x=VAL "60045": GO SUB VAL "950"
240 LET i=VAL "12": PRINT AT i,NOT PI;"Finish at line? (last)":
GO SUB VAL "6e3": IF NOT LEN a$ THEN LET b=a-c: GO TO VAL "250"
245 LET b=VAL "64*INT VAL a$-c"
250 RANDOMIZE USR VAL "59806": RANDOMIZE USR FN p(VAL "62472")
252 LET i=VAL "14": LET j=NOT PI: LET a$="Left hand margin = ":
LET x=VAL "60927": GO SUB VAL "850"
255 LET i=VAL "16": PRINT AT i,NOT PI;"Number of copies (1)": G
O SUB VAL "6e3": IF NOT LEN a$ THEN LET a$="1"
256 LET j=VAL a$
260 CLS : PRINT AT VAL "20",NOT PI;"Press the q key to quit pri
nting"
270 LET x=VAL "60049": GO SUB VAL "950"
275 LET c=PEEK VAL "62470": IF c THEN LPRINT CHR$ c
280 RANDOMIZE USR VAL "60038"
285 LET c=PEEK VAL "62471": IF c THEN LPRINT CHR$ c
290 RANDOMIZE USR VAL "59806": GO TO VAL "10"
300 CLS : PRINT "Printer control graphics chars:"
305 LET b=VAL "4": FOR i=NOT PI TO VAL "15": LET b=ABS (b-VAL "
4"): PRINT AT i+VAL "4",b;i+VAL "128":CHR$ (i+VAL "128")
307 FOR j=NOT PI TO INT PI: LET c=PEEK VAL "60860+i*4+j": LET a
$=STR$ c: IF c=VAL "255" THEN LET a$=""
308 PRINT AT i+VAL "4",VAL "10+4*j";a$: NEXT j: NEXT i
320 INPUT "Type graphics character code 128-143 (ENTER if finis
hed): "; LINE a$
325 IF NOT LEN a$ THEN GO TO VAL "400"
340 LET b=VAL a$: IF b<VAL "128" OR b>VAL "143" THEN GO TO VAL
"320"
350 PRINT AT VAL "21",INT PI: FLASH SGN PI;b;: PRINT " ";CHR$ b
355 FOR j=NOT PI TO INT PI: POKE VAL "60348+b*4+j",VAL "255": N
EXT j
360 FOR j=NOT PI TO INT PI: INPUT "Code? (ENTER if finished): "
; LINE a$: IF NOT LEN a$ THEN GO TO VAL "300"
370 POKE VAL "60348+b*4+j",VAL a$: PRINT AT VAL "21",VAL "10+4*
j";VAL a$: NEXT j: GO TO VAL "300"
400 CLS : PRINT AT INT PI,NOT PI;"Reset interface/printer codes
?": LET i=VAL "5": GO SUB VAL "920": CLS : IF NOT i THEN GO TO V
AL "25"
401 PRINT AT INT PI,NOT PI;"just ENTER to keep values given:"
403 LET j0=VAL "27": LET i=VAL "8": LET j=NOT PI: LET a$="Inter
face control code1=": LET x=VAL "60924": GO SUB VAL "850"
404 LET i=VAL "9": LET j=VAL "18": LET a$="code2=": LET x=VAL "
62470": GO SUB VAL "850"
405 LET i=VAL "10": LET j=VAL "18": LET a$="code3=": LET x=VAL
"62471": GO SUB VAL "850"
406 LET i=VAL "11": PRINT AT i,VAL "18";"code4=";FN p(VAL "6247
2"): LET i=VAL "12": LET j0=VAL "24": GO SUB VAL "6e3": LET j0=V
AL "27": IF LEN a$ THEN LET b=VAL a$: LET x=VAL "62472": GO SUB
VAL "950"
410 LET i=VAL "14": LET j=NOT PI: LET a$="Printer carriage retu
rn=": LET x=VAL "60925": GO SUB VAL "850"
420 LET i=VAL "16": LET j=NOT PI: LET a$="Printer linefeed=": L
ET x=VAL "60926": GO SUB VAL "850"
430 LET i=VAL "18": LET j=NOT PI: LET a$="Left margin on printi
ng=": LET x=VAL "60927": GO SUB VAL "850"
490 GO TO VAL "20"
500 INPUT "": PRINT #NOT PI;" press the "; FLASH SGN PI;"ENTER"

```

```

; FLASH NOT PI;" key to proceed"" press "; FLASH SGN PI;"c"; F
LASH NOT PI;" to change the choice "
510 LET a$=INKEY$: IF a$="c" OR a$="C" THEN GO TO VAL "20"
520 IF CODE a$=VAL "13" THEN GO TO VAL "600"
530 GO TO VAL "510"
600 IF b=VAL "116" THEN GO TO VAL "700"
610 IF b=VAL "121" THEN CLS : GO TO VAL "10"
620 IF b=VAL "115" THEN CLS : GO TO VAL "1e3"
630 IF b=VAL "109" THEN GO TO VAL "2e3"
640 IF b=VAL "106" THEN LET a=USR VAL "59081": LET a=NOT PI: GO
TO VAL "2e3"
650 IF b=VAL "112" THEN GO TO VAL "200"
660 IF b=VAL "103" THEN GO TO VAL "300"
670 IF b=VAL "100" THEN GO TO VAL "5e3"
680 IF b=VAL "101" THEN GO TO VAL "1500"
699 CLS : STOP
700 CLS : PRINT "SAVE TASWORD"" : POKE VAL "23777",NOT PI: CAT
#"a": SAVE #"tasword" LINE VAL "15": VERIFY #"tasword": CAT #:
SAVE #"tascode",VAL "54784",VAL "10751": POKE VAL "23767",NOT P
I: CLS : GO TO VAL "25"
850 PRINT AT i,j;a$:PEEK x: GO SUB VAL "6e3": IF LEN a$ THEN PO
KE x,VAL a$
860 RETURN
930 LET i=INKEY$="y" OR INKEY$="Y": IF INKEY$="n" OR INKEY$="N"
OR i THEN RETURN
940 GO TO VAL "930"
950 POKE x,b-VAL "256*INT (b/256)": POKE x+SGN PI,INT (b/VAL "2
56"): RETURN
1000 CLS : PRINT "SAVE TEXT FILE"
1010 LET b=FN p(VAL "62216"): GO SUB VAL "8e3": GO SUB VAL "8100
": SAVE #a$,b,a: GO TO VAL "20"
1500 CLS : PRINT "ERASE FILE"
1510 GO SUB VAL "8e3": GO SUB VAL "8100": ERASE #a$: GO TO VAL "
20"
2000 CLS : PRINT "LOAD/MERGE TEXT FILE"
2010 LET b=FN p(VAL "62216"): LET b=a+b: GO SUB VAL "8e3": GO SU
B VAL "8100": LOAD #a$,b: GO TO VAL "10"
3000 FOR i=VAL "23296" TO VAL "23361": POKE i,VAL "32": NEXT i
3005 POKE VAL "23362",NOT PI
3010 PRINT AT VAL "8",NOT PI;"type word to be replaced / found"
3012 LET j0=NOT PI: LET i=VAL "10": GO SUB VAL "6e3": IF NOT LEN
a$ THEN GO TO VAL "10"
3020 LET j=NOT PI: FOR i=SGN PI TO LEN a$: POKE VAL "23297"+i,CO
DE a$(i): IF a$(i)=" " THEN LET j=j+SGN PI
3021 NEXT i
3022 IF j THEN CLS : PRINT AT VAL "12",NOT PI;"just a word - no
spaces allowed": GO TO VAL "3e3"
3025 POKE VAL "23297",LEN a$
3030 PRINT AT VAL "12",NOT PI;"with (just ENTER for find only)":
LET i=VAL "14": GO SUB VAL "6e3"
3040 IF NOT LEN a$ THEN POKE VAL "23362",SGN PI: GO TO VAL "3060
"
3050 FOR i=SGN PI TO LEN a$: POKE VAL "23329"+i,CODE a$(i): NEXT
i
3060 LET a=USR VAL "64955": LET a=USR VAL "64333": GO TO VAL "20
"
4000 PRINT "TASWORD 2 Tasman Software 1983"" : RETURN
5000 CLS : PRINT "DRIVE DIRECTORY": GO SUB VAL "8e3": PRINT #NOT
PI;"Press a key to continue..": PAUSE NOT PI: GO TO VAL "20"
6000 LET a$="": PRINT AT i,j0: FLASH SGN PI;" "

```

```

6010 LET j=j0: IF LEN INKEY$ THEN GO TO VAL "6010"
6020 LET b$=INKEY$
6030 IF NOT LEN b$ THEN GO TO VAL "6020"
6040 IF CODE b$=VAL "13" THEN PRINT AT 1,j;" ": RETURN
6050 IF CODE b$<>VAL "12" THEN GO TO VAL "6170"
6060 IF j=j0 THEN GO TO VAL "6200"
6070 LET j=j-SGN PI: PRINT AT 1,j; FLASH SGN PI;" "; PRINT " ";
  LET a$=a$( TO j-j0); GO TO VAL "6200"
6170 IF CODE b$<VAL "32" OR CODE b$>VAL "127" THEN GO TO VAL "62
00"
6180 BEEP VAL ".005",VAL "5": PRINT AT 1,j;b$; FLASH SGN PI;" ";
  LET j=j+SGN PI: LET a$=a$+b$
6190 IF j=VAL "32" THEN PRINT AT 1+SGN PI,NOT PI;" ": RETURN
6200 IF LEN INKEY$ THEN GO TO VAL "6200"
6210 GO TO VAL "6020"
7000 DEF FN p(x)=PEEK x+VAL "256*PEEK (x+1)"
8000 PRINT #NOT PI;"Which drive (a/b):"
8005 GO SUB VAL "8200"
8010 LET d$=INKEY$: IF NOT LEN d$ THEN GO TO VAL "8010"
8020 LET d$="b:" AND (d$="B" OR d$="b"); CAT *d$: RETURN
8100 POKE VAL "23658",VAL "8": INPUT "Enter file name: "; LINE a
$: IF NOT LEN a$ OR LEN a$>VAL "10" THEN GO TO VAL "8100"
8110 LET a$=d$+a$: POKE VAL "23658",NOT PI: RETURN
8200 IF LEN INKEY$ THEN GO TO VAL "8200"
8210 RETURN

```

VU-FILE & VU-CALC ON WAFADRIE

Psion's VU-FILE and VU-CALC programs are two of the most widely used Spectrum utility programs. Unfortunately, since their machine code portions locate low down in memory (at around address 25000), they are not easy to transfer to wafer (BASIC starts at around 26000 when the Wafadrive is initialised). Both programs require BASIC to be left intact as many of their functions are carried out from BASIC.

VU-FILE and VU-CALC are interfaced to the Wafadrive by means of a machine code "wedge" program. Although the programs are different in each case, the method is the same and is described below. No knowledge of machine code is necessary but make sure that the steps are followed exactly:

1. Have ready a (blank) 16K wafer for the final version of the program and another wafer which has at least 12K of spare capacity. These will henceforth be referred to as the "program" and "working" wafers respectively.
2. Starting with the computer reset, initialise the Wafadrive and then enter the appropriate loader program listed on one of the accompanying sheets. Check it carefully and don't be tempted to make any of your own "improvements" at this stage.
3. With the Wafadrive initialised, and with the working wafer in drive A, save the loader: `SAVE "name" LINE 8e3`, where "name" is "vu file" or "vu calc" as appropriate.
4. Enter `NEW #` to clear out this first program and then type in the relevant "converter" program. These are longer so you will have to be more careful with your checking. Take particular care with the numbers in the DATA statements. Save the program: `SAVE "convert"`. The DATA statements can be checked by entering `GO SUB 8e3` in direct mode. If you have made a mistake the program will tell you which line is faulty.
5. Now you can RUN the converter program. The first thing you will be asked to do is to play the program cassette. You can cue the tape up to the beginning of the code file if you like ("C" or "t") - this is the only file that the converter programs use.
6. Once loaded the converter program adds the machine code interface program, splits the program code into two sections and saves them onto the working wafer. It also then saves a copy of itself just in case you forgot to do so.
7. When this has been done a few instructions will be displayed then the program will be loaded from the working wafer and run.
8. Now make a "proper" copy of the program onto the program wafer - don't use the version on the working wafer but keep it as a backup. Exit to BASIC (in the case of VU-FILE this can be done by selecting "load" and removing both wafers to generate an error), insert the program wafer in drive A and enter `GO TO 9999`. The wafer will be formatted and the final version of

VU-FILE/VU-CALC saved to it. Other copies can be made at any time using this method.

To use the wafer versions of VU-FILE/VU-CALC simply insert the wafer in drive A and enter LOAD *. The screen display has been dropped - it only adds to loading time and would not stay on the screen for very long anyway. You can add it if you like though.

I hope you will agree that the load and save options in the new versions of the programs are pleasant to use. You can use the same techniques in your own programs. Note that in both cases the directory can be examined by selecting the "load" option and entering a null file name to effect a return to the program.

The VU-FILE BASIC loader is written assuming a Centronics printer is to be used. If you have a "ZX" type printer, omit the OPEN # statement in line 8000. If you have an RS-232 printer replace it with OPEN #INT PI,"r" and set the baud rate if required - eg. by following it with FORMAT #"r",VAL "9600". VU-CALC performs its printing via COPY and without further modification can only be used with a "ZX" type printer. Do not press the BREAK key whilst VU-CALC is printing.

In the BASIC loaders you will notice the use of VAL to represent all numbers, and expressions containing numbers, and things like NOT PI (to represent 0), SGN PI (1) and INT PI (3). In case you are not aware, these are memory saving devices (VAL "9600" saves 3 bytes and INT PI saves 5).

The VERIFY and CAT # statements in line 9999 cause the files on the wafer to be recorded on the tape such that loading time is minimized - again you can use this technique yourself.

If you crash either of the programs and get a BASIC error message simply use RUN to restart them. GO TO 9000 can still be used with VU-CALC to get a report of the faulty cell.

Cassette files can be loaded by modifying the LOAD * commands in line 2000 (VU-FILE)/1000 (VU-CALC). Change them back to the cassette syntax and drop the GO SUB call. The files can then be loaded from tape and saved to wafer - try this with the example files supplied on the program cassettes.

The conversion to wafer does mean that there is less room for your files in memory but this should not be a problem as the capacity is still generous. Loading very large files from cassette could crash the programs so be careful.

Finally, the VU-FILE and VU-CALC programs are copyright and any copying must be in accordance with the appropriate laws. Please do not ask ROTRONICS to convert programs for you - we cannot do so. We cannot make any guarantees as to the success of the conversions and there may be more than one version of the programs to trap the unwary. We do welcome any comments or suggestions you may care to make.

VU-FILE LOADER PROGRAM:

```

100 GO SUB VAL "7e3": GO TO USR a
1000 CLS : PRINT "Save a file": GO SUB VAL "6e3": GO TO USR a
1010 SAVE #g$,s,1: GO TO USR a
2000 CLS : PRINT "Load a file": GO SUB VAL "6e3": LOAD #g$,s: GO
  TO USR a
3000 LPRINT f$( TO VAL "32"): GO TO USR a
6000 PRINT #NOT PI;"Which drive? (A/B): "
6010 LET d$=INKEY$: IF NOT LEN d$ THEN GO TO VAL "6010"
6020 LET d$="b:" AND (d$="B" OR d$="b"): CAT #d$: INPUT "Enter f
ile name: "; LINE g$: IF NOT LEN g$ THEN RUN
6030 LET f$( TO VAL "10")=g$: LET s=VAL "CODE f$(11)+256*CODE f$
(12)": LET l=VAL "CODE f$(13)+256*CODE f$(14)": LET g$=d$+g$: RE
TURN
7000 DIM f$(VAL "32"): LET a=VAL "63078": RETURN
8000 POKE VAL "23732",VAL "93": POKE VAL "23733",VAL "246": CLEA
R VAL "27379": LOAD #"vu code1",VAL "63070": LOAD #"vu code2",VA
L "27380": OPEN #INT PI,"c": BORDER SGN PI: GO SUB VAL "7e3": G
O TO USR VAL "63070"
9999 FORMAT #"a:VUFILE."+STR$ PEEK VAL "23672": CAT #"a:": SAVE
#"vu file" LINE VAL "8e3": VERIFY #: CAT #: SAVE #"vu code1",VAL
"63070",VAL "2462": VERIFY #"vu code1": CAT #: SAVE #"vu code2"
,VAL "27380",VAL "3350": CAT *

```

VU-FILE CONVERTER PROGRAM:

```

10 REM VU-FILE Converter
200 CLEAR 4e4: GO SUB 8e3: PRINT "Rewind the VU-FILE tape, and
  start the recorder ready to loadin the code file:": GO SUB 7e
3: LOAD "C"CODE 5e4
300 CLS : PRINT "Ensure that the working wafer isin drive A rea
dy to accept the code files:": GO SUB 7e3
310 SAVE #"a:vu code1",49830,2462: SAVE #"a:vu code2",52292,335
0: SAVE #"a:convert"
400 CLS : PRINT "When this program has finished, VU-FILE will l
oad and run - provided all is correct that is.""Exit to BA
SIC, remove your working wafer and insert a 16K wafer in d
rive A ready to be formatted and have VU-FILE savedon it.""
When you are sure all is correctenter GO TO 9999 to save the new
version of VU-FILE.": GO SUB 7e3
410 LOAD #"a:vu file": STOP
7000 REM Press a key
7010 INPUT "": PRINT #0;"Press a key when ready...": PAUSE 0: CL
S : RETURN
8000 REM POKE DATA
8010 DATA "205","108","246","205","25","116","24","6","205","108
","246","205","89","116","197","62","2163"
8020 DATA "9","205","184","246","205","202","246","33","182","92
","126","254","167","32","25","229","2437"
8030 DATA "205","197","246","225","1","244","8","205","232","25"
,"17","255","97","213","205","214","2589"
8040 DATA "246","209","19","205","192","246","24","27","33","0",
"98","205","197","246","17","243","2207"
8050 DATA "106","205","214","246","33","181","92","1","244","8",
"229","205","85","22","209","19","2099"
8060 DATA "205","192","246","62","15","205","184","246","193","2
01","33","141","92","119","205","175","2514"
8070 DATA "13","201","33","0","64","24","11","17","8","247","24"
,"6","33","8","247","17","953"

```

```

9080 DATA "0","64","1","244","8","237","176","201","237","83","1
78","92","237","115","0","247","2120"
8090 DATA "237","75","0","247","42","61","92","35","229","183","
237","66","68","77","3","235","1887"
8100 DATA "54","62","235","27","27","27","237","83","61","92","1
9","225","237","184","19","235","1824"
8110 DATA "249","201","0","0","0","0","0","0","0","0","0","0"
,"0","0","0","450"
8200 PRINT "Wait a while..."
8300 RESTORE 8e3: LET a=49830: FOR m=1 TO 11: LET s=0: FOR n=1 T
O 16: READ a$: LET s=s+VAL a$: POKE a,VAL a$: LET a=a+1: NEXT n:
  READ a$: IF VAL a$<>s THEN PRINT "CHECKSUM ERROR IN LINE ";8e3+
10*m: STOP
8310 NEXT m: CLS : RETURN

```

VU-CALC LOADER PROGRAM:

```

10 GO TO USR VAL "63058"
1000 GO SUB VAL "6e3": LOAD *a$: RUN
3000 GO SUB VAL "7e3": PRINT AT VAL "9",VAL "2": "ENTER 1 : EXIT
PROGRAM" TAB VAL "9": "2 : CLEAR WORKSHEET" TAB VAL "9": "3 : R
ETURN TO VU-CALC": INPUT "OPTION? ";a: GO TO VAL "3e3+100*a*(a>0
AND a<4)"
3100 GO SUB VAL "7e3": STOP
3200 CLEAR : DIM b$(VAL "100"): DIM c$(VAL "20"): GO TO USR VAL
"63050"
3300 RUN
4000 GO SUB VAL "6e3": SAVE *a$,VAL "30457",VAL "PEEK 30466+256*
PEEK 30467-30457": RUN
5000 LET c$=STR$ VAL b$: GO TO USR VAL "63066"
6000 INPUT "": PRINT #NOT PI;"Which drive (A/B): "
6010 LET d$=INKEY$: IF NOT LEN d$ THEN GO TO VAL "6010"
6020 LET d$="b:" AND (d$="B" OR d$="b"): CAT *d$: INPUT "Enter f
ile name: "; LINE a$: IF NOT LEN a$ THEN RUN
6030 LET a$=d$+a$: RETURN
7000 CLS : PRINT TAB VAL "13": "VU-CALC": RETURN
8000 BORDER SGN PI: POKE VAL "23732",VAL "73": POKE VAL "23733",
VAL "246": CLEAR VAL "27523": LOAD *"v/c 1",VAL "63050": LOAD *"
v/c 2",VAL "27524": GO TO VAL "3200"
9000 CLS : PRINT "" "ERROR was at" "" "COLUMN ";PEEK VAL "28784"+
SGN PI
9100 LET a=PEEK VAL "28785": LET b=INT VAL "a/26": LET a=a-b*VAL
"26": LET d$=CHR$ VAL "b+64"+CHR$ VAL "a+65": IF NOT b THEN LET
d$=d$(VAL "2")
9200 PRINT "ROW ";d$: PRINT #NOT PI;"Press ENTER to continue"
: PAUSE NOT PI: RUN
9999 CLEAR : FORMAT *"a:VUCALC."+STR$ PEEK VAL "23672": CAT #"a:
": SAVE *"vu calc" LINE VAL "8e3": VERIFY *: CAT #: SAVE *"v/c 1
",VAL "63050",VAL "2482": VERIFY *"v/c 1": CAT #: SAVE *"v/c 2",
VAL "27524",VAL "2978": CAT *

```

VU-CALC CONVERTER PROGRAM:

```

10 REM VU-CALC Converter
200 CLEAR 4e4: GO SUB 8e3: PRINT "Rewind the VU-CALC tape, and
  start the recorder ready to loadin the code file:": GO SUB 7e
3: LOAD "t"CODE 5e4
300 CLS : PRINT "Ensure that the working wafer isin drive A rea
  dy to accept the code files:": GO SUB 7e3
310 SAVE #"a:v/c 1",49810,2482: SAVE #"a:v/c 2",52292,2978: SAV
  E #"a:convert"
400 CLS : PRINT "When this program has finished, VU-CALC will l
  oad and run - provided all is correct that is."'"Exit to BA
  SIC, remove your working wafer and insert a 16K wafer in d
  rive A ready to be formatted and have VU-CALC savedon it.'"
  When you are sure all is correctenter GO TO 9999 to save the new
  version of VU-CALC.": GO SUB 7e3
410 LOAD *"a:vu calc": STOP
7000 REM Press a key
7010 INPUT "": PRINT #0;"Press a key when ready...": PAUSE 0: CL
  S : RETURN
8000 REM POKE DATA
8010 DATA "205","96","246","205","68","100","24","14","205","96"
  ,"246","205","71","100","24","6","1911"
8020 DATA "205","96","246","205","92","117","33","208","7","183"
  ,"237","66","32","10","205","172","2114"
8030 DATA "14","33","96","246","229","195","74","100","197","62"
  ,"63","205","190","246","205","207","2362"
8040 DATA "246","33","182","92","126","254","167","32","25","229"
  ,"205","202","246","225","1","244","2509"
8050 DATA "8","205","232","25","17","143","98","213","205","219"
  ,"246","209","19","205","197","246","2487"
8060 DATA "24","27","33","144","98","205","202","246","17","131"
  ,"107","205","219","246","33","181","2118"
8070 DATA "92","1","244","8","229","205","85","22","209","19","2
  05","197","246","62","56","205","2085"
8080 DATA "190","246","193","201","33","141","92","119","195","1
  75","13","33","0","64","24","11","1730"
8090 DATA "17","8","247","24","6","33","8","247","17","0","64","
  1","244","8","237","176","1337"
8100 DATA "201","237","83","178","92","237","115","5","247","237"
  ,"75","5","247","42","61","92","2154"
8110 DATA "35","229","183","237","66","68","77","3","235","54","
  62","235","27","27","27","237","1802"
8120 DATA "83","61","92","19","225","237","184","19","235","249"
  ,"201","0","0","0","0","0","0","1605"
8200 PRINT "Wait a while..."
8300 RESTORE 8e3: LET a=49810: FOR m=1 TO 12: LET s=0: FOR n=1 T
  O 16: READ a$: LET s=s+VAL a$: POKE a,VAL a$: LET a=a+1: NEXT n:
  READ a$: IF VAL a$(<>s) THEN PRINT "CHECKSUM ERROR IN LINE ";8e3+
  10*m: STOP
8310 NEXT m: CLS : RETURN

```